УДК 681.51:629 DOI 10.21685/2307-4205-2020-3-4

А. А. Тюгашев

ПРОБЛЕМЫ НАДЕЖНОГО ИНТЕЛЛЕКТУАЛЬНОГО УПРАВЛЕНИЯ СЛОЖНЫМИ СИСТЕМАМИ В РЕЖИМЕ РЕАЛЬНОГО ВРЕМЕНИ

A. A. Tyugashev

PROBLEMS OF RELIABLE INTELLIGENT MANAGEMENT OF COMPLEX SYSTEMS IN REAL TIME

Аннотация. В самых разных сферах человечество использует сегодня сложные технические системы. Это автоматизированные производства, космические аппараты, атомные станции и пр. Подобные системы состоят из множества приборов, датчиков, исполнительных механизмов, объединенных в подсистемы различного назначения. Для выполнения системой поставленных перед ней целевых задач необходимо обеспечить согласованное во времени и по логике управления функционирование входящих в ее состав устройств. Средства управления должны реализовывать надежное и гибкое «интеллектуальное» управление в различных смыслах, анализируемых в статье. Описывается применение логики управляющих алгоритмов реального времени, ранее предложенной автором, для случая ограниченных доступных ресурсов. Также описаны специальные инструментальные программные средства поддержки верификации и синтеза необходимых средств управления.

Ключевые слова: интеллектуальное управление, управление сложными техническими комплексами, управляющий алгоритм реального времени, искусственный интеллект, верификация программ, формальные методы.

Abstract. The complex technical systems are used today in various areas. We can name transportation systems, automated manufactures, nuclear power plants, etc. Such system consists of subsystems with different objectives. Each subsystem includes dozens of devices, sensors, actuators, etc. To achieve the system's goals, the control means must provide well coordinated functioning of all these devices. We need the right coordination in time (real-time mode) and due to logical and physical reasons. The modern control should guarantee the completion of the system tasks even in case of some abnormal situations and unexpected external events. The paper is devoted to the definition of «intelligent real-time control for complex systems» and analysis of the most important features of needed control means. The paper also describes the application of real-time control algorithms logic for the systems with limited available resources. We present also some software tools for support of verification and synthesis of the complex system's control logic.

Keywords: intelligent control, management of complex technical complexes, real-time control algorithm, artificial intelligence, program verification, formal methods.

Введение

В XXI в. человечество использует значительное число сложных технических комплексов и систем. Автоматизированные производства, атомные электростанции, космические корабли могут быть названы в качестве характерных примеров [1–4].

Что мы вкладываем в действительности в понятие «сложной» системы? Во-первых, мы ожидаем увидеть наличие комплексной структуры с многими уровнями, часто – иерархически организованную. Иными словами, комплекс состоит из подсистем, которые в свою очередь включают в свой состав десятки датчиков, приборов, исполнительных механизмов и пр.

Во-вторых, мы подразумеваем сложное поведение, часто – в меняющемся внешнем окружении. Задача обеспечения подобного комплекса средствами гибкого и надежного управления является весьма нетривиальной. Здесь встает очередной вопрос: что понимать под «гибким» и «надежным» управлением?

Каждая техническая система создается для выполнения определенных функций – транспортировки пассажиров и грузов, выработки электроэнергии, производства изделий и пр. Соответственно, у нее будет наличествовать набор целей (функциональных задач).

Весьма важным аспектом многих современных сложных технических систем является функционирование в реальном режиме времени. Во многих случаях выполнение поставленных задач привязывается тем или иным способом к временной шкале. При этом задается пара «целевое состояние», «время». Нередко система должна обеспечить не просто достижение абстрактных «целей», а набор логически скоординированных и привязанных к определенным моментам времени действий. Например, для достижения главной цели необходимо предварительно реализовать выполнение набора подготовительных операций, тоже привязанных к шкале времени. Другие необходимые операции могут потребоваться после достижения главной цели (например, можно упомянуть чистку салона самолета по завершении перелета) [2, 4].

Системы реального времени можно разделить на реагирующие и «активные». Если моменты времени достижения поставленных целей изначально заданы, система имеет «активный характер», в отличие от пассивных «реагирующих» систем, находящихся в цикле ожидания внешних событий. Иногда действие имеет ненулевую продолжительность, представляя собой протяженный во времени процесс. Многие из процессов должны исполняться при этом параллельно. Для некоторых из функциональных процессов имеет значение синхронизация моментов их начала и окончания – способ организации параллельного выполнения. Во многих ситуациях, в том числе по соображениям безопасности, запрещено наложение определенных процессов друг на друга. К сожалению, основные известные модели в области моделирования поведения и управления сложными системами посвящены реагирующим системам.

А. А. Калентьевым предложена и автором настоящей статьи развита модель алгебры и логики управляющих алгоритмов реального времени, напротив, подходящая для описания поведения «активных» систем [3, 5].

Для нашего случая сложная техническая система, состоящая из различных приборов, датчиков и агрегатов, должна выполнить некоторый заданный план действий с привязкой к временной шкале (план-график). На самом деле имеется обычно не единственный план, а набор планов для различных вариантов развития событий, например, один для полностью штатного функционирования аппаратуры, другой – в случае возникновения одного из возможных отказов, третий – другого отказа, и т.д. Подобный план в аэрокосмической отрасли принято называть «циклограммой», представление о нем можно получить, глядя на рис. 1. На нем показан факт, что выполнение некоторого процесса не обязательно, а зависит от складывающейся ситуации, отражен цветом.

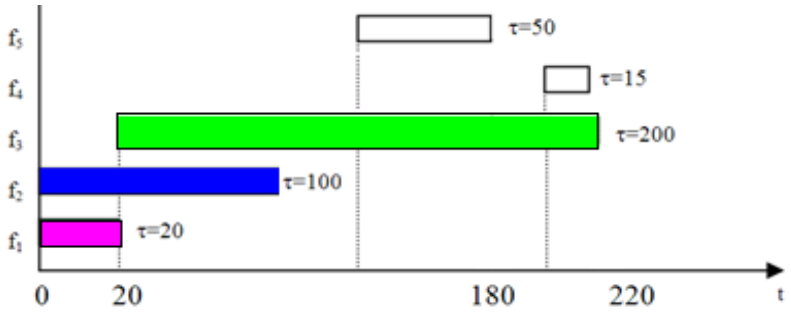

Рис. 1. Представление плана функционирования сложной системы

Для адекватного управления в условиях изменяющейся внешней среды и, возможно, наступления непредсказуемых событий, в системе должны быть заложены возможности гибкого реагирования на подобные ситуации. Должны при этом быть задействованы возможности различных входящих в состав системы приборов и устройств, основанных на применении разных физических принципов и явлений [6–8].

При этом следует иметь в виду, что управление современными сложными техническими комплексами реализуется на базе применения компьютеров, нередко – бортовой вычислительной сети. Реализация системного плана действий непосредственно выполняется управляющим программным обеспечением, реализующим логику управления, с внесением в нее необходимых вариантов.

В этой ситуации имеется еще одно следствие. Как известно, система, состоящая из бóльшего числа элементов, обычно менее надежна, чем менее сложная (за исключением специальных случаев, когда усложнение специально вносится для повышения надежности, например, при использовании

дублирования). Получается, что говоря о надежном управлении, мы должны принимать во внимание и это обстоятельство.

Сложная система при надежном управлении должна обеспечить, как говорилось выше, выполнение поставленных задач даже при возникновении нештатных ситуаций, отказах частных приборов и пр. Достигают этого путем использования избыточности. Избыточность может быть структурной, когда важные приборы дублируются, и функциональной. В случае функциональной избыточности подразумевается, что прибор может выполнить дополнительные функции, взяв на себя обязанности, возложенные на иное устройство, возможно, с ухудшением точности или качества. Например, в системе ориентации космического корабля возможно использование разных видов датчиков – Солнца, Земли, звезд [6, 7]. Возможна ее разновидность – временная избыточность, когда функция выполняется, но медленнее, чем при штатной работе.

Для эффективного использования избыточности нам необходима заложенная еще на стадии проектирования в систему возможность реконфигурации. Реконфигурация в случае переключения на дублирующий комплект оборудования должна производиться путем выдачи специальных управляющих воздействий со стороны системы управления, выявляющей ее необходимость путем анализа показаний датчиков, измерений параметров и пр. В более сложной ситуации «интеллект» системы управления должен обладать знаниями о возможности замещения функций одного прибора другим, определять неисправности и подбирать возможные варианты парирования [6, 8].

В любом случае для выполнения поставленных задач будут необходимы определенные виды функциональности на заданных временных промежутках эксплуатации сложной системы.

Можно представить «сумму» того или иного вида необходимой функциональности, формируемую различными приборами, обладающими подобными возможностями. При этом функциональность прибора может зависеть от его текущего режима работы.

Анализ аномальных ситуаций и выполнение парирующих действий выполняются управляющим программным обеспечением, от коего непосредственно зависит достижение поставленных перед техническим комплексом целей. Подобное программное обеспечение (ПО) принято называть «критически важным» (‘Mission-Critical Software’) [3, 5]. Например, ПО может выдавать на бортовую аппаратуру команды, переводимые в электрические импульсы. Примеры подобных команд:«Немедленно активировать устройство № 2 системы № 1» или «Отключить второй силовой гиродин» [8, 9].

Переключение режимов работы устройств приводит к изменению уровней потребляемых ими ресурсов. Наиболее очевидным здесь является потребление электричества. Любой прибор как минимум будет иметь два состояния – «ВКЛ» и «ОТКЛ», в котором потребление равно нулю.

Могут быть и иные виды ресурсов, потребляемых при функционировании оборудования. При этом часть из этих ресурсов восполняемые (например, солнечные батареи вырабатывают, а не потребляют электрическую энергию), а некоторые – невосполняемые (например, запас ракетного топлива в баках космического аппарата).

В ходе функционирования сложной системы постоянно происходит переключение разных бортовых устройства из режима в режим и, соответственно, изменяются суммарные уровни потребления различных ресурсов. При этом существует максимально доступный уровень – «красная линия» – для каждого из видов ресурсов, который нельзя превышать.

Для успешной работы в описанных выше условиях необходимо, чтобы средства управления имели целый ряд важных особенностей. Мы должны иметь внутреннее представление о текущем состоянии самой управляемой системы (со средствами описания уровня функциональности/работоспособности оборудования) и ее окружения. Мы, помимо этого, должны иметь представление о целях с пониманием того, какие из них уже сделаны, а какие ждут исполнения в какие моменты времени.

Таким образом, выделим важнейшие черты подобной системы:

– непредсказуемая внешняя среда, представление о которой должно присутствовать в системе управления;

– разнообразное входящее в состав системы оборудование, организованное по иерархическому принципу;

– необходимость достижения набора поставленных перед системой целей;

– работа в режиме реального времени;

– система управления на базе современных цифровых ЭВМ;

– использование специального управляющего программного обеспечения, задающего текущую конфигурацию системы и режимы работы бортовых устройств;

– возможные отказы оборудования должны быть надлежащим образом скомпенсированы, а стоящие перед системой цели – достигнуты в любом случае;

– ограниченные уровни доступных ресурсов.

Таким образом, нам необходимо сначала построить адекватные модели для представления сложной системы с перечисленными чертами и средств управления ею.

Моделирование управления сложной системой с ограниченными ресурсами

Для управления сложной технической системой в настоящее время используется специальное управляющее программное обеспечение (ПО) – комплекс управляющих алгоритмов реального времени [2, 3, 6]. Как правило, управляющий алгоритм реализуется соответствующим программным модулем. При этом в сложных современных системах выполнение модулей бортового программного обеспечения происходит в мультипрограммном режиме – параллельно.

Управляющий программный модуль может содержать различные операторы, но для нас важны следующие действия: 1) вызов другого модуля комплекса бортовых программ; 2) выдача команды управления на бортовую аппаратуру; 3) установка или сброс логического условия (признака, флага). Данные флаги влияют на реализацию системного плана. Например, флаг может означать «уровень заряда аккумуляторной батареи № 1 слишком низкий» или «основной двигатель вышел на рабочий режим». Полный набор подобных флагов формирует «вектор текущего состояния» сложной системы (ВТС), или «логический вектор». Подмножества ВТС могут рассматриваться как частные векторы состояния, например, определенной подсистемы [10]. Текущие значения логических условий, формирующих вектор состояния, определяют возможные сценарии исполнения управляющих алгоритмов, иными словами, какая из ветвей логики управления будет использована.

Различные сценарии исполнения управляющих программ задают разные привязанные по времени (таймированные) последовательности операций – разные варианты реализации системного плана или циклограммы (см. рис. 1). Как уже отмечалось, на самом деле мы должны реализовать набор вариантов системного плана с учетом возможных непредвиденных ситуаций и парирования отказов оборудования. На одной диаграмме операции и процессы, принадлежащие различным сценариям, могут быть выделены, например, цветом.

Дополнительная сложность реализации логики управления в данном случае обусловлена следующим обстоятельством. Каждый из управляющих алгоритмов, выполняемых параллельно, выполняет путем выдачи команд бортовым устройствам свои цепочки операций. При этом каждый из модулей управляющего ПО может запускать несколько других, которые в свою очередь – еще модули, и т.п. Мы должны учитывать дерево вызовов управляющих алгоритмов: $УА_1 \rightarrow УА_2 \rightarrow УА_3 \ldots$ (рис. 2).

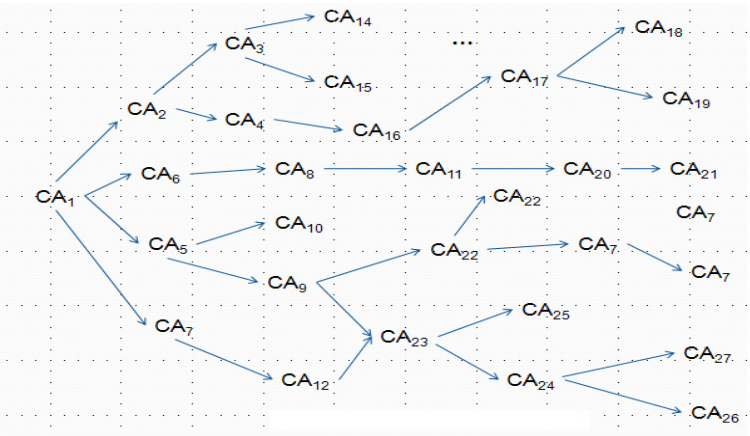

Рис. 2. Дерево вызовов управляющих программных модулей

В данных условиях человеку, планирующему логику управления сложной системой, практически невозможно полностью удерживать в голове все взаимосвязи между управляющими про-

граммными модулями и параллельно реализуемые ими цепочки действий, а также зависимости выполняемых действий от комбинаций логических условий – частных векторов состояния.

Еще одной проблемой становится проверка «устойчивости» или «согласованности» логики управления с точки зрения контроля отсутствия превышения максимально допустимых уровней потребления бортовых ресурсов на всем промежутке выполнения системой целевых задач и при всех возможных сценариях реализации логики управления.

Введем в рассмотрение следующие множества для моделирования сложной системы с данными характеристиками и ее средств управления [9]:

$EU = \{EU_i\}$, $i = 1...N$ – множество устройств, входящих в состав системы;

$FA = \{FA_j,\}$ $j = 1...P$ – множество видов предоставляемой устройствами функциональности;

$AR = \{AR_j,\}$ $j = 1...M$ – множество необходимых ресурсов;

$Lim\ (AR_j) = \{\ (LimAR_1, LimAR_2, ..., LimAR_M)\ \}$ – набор максимально доступных уровней бортовых ресурсов;

$MD\ (BA_i) = \{\ (MD_{i1}, MD_{i2}... MD_{iN})\ \}$ – множество режимов работы аппаратных средств;

$CL\ (MD_{ik}) = \{\ (\ CL_{i1}, CL_{i2}... CL_{iN}\ )\ \}$ – множество векторов потребления данным устройством в указанном режиме различных видов бортовых ресурсов.

Проблема синтеза логики управления сложной системой

Каким образом мы можем описать логику управления представленной сложной системой в реальном времени? Обычно, говоря о логике, мы имеем в виду использование набора аксиом и правил вывода (рассуждений). Но что конкретно мы должны применить в данном случае? В качестве аксиом могут рассматриваться задания параметров базисных процессов (длительность, время начала), характеристики бортовых устройств (предоставляемая функциональность, потребление ресурсов в разных режимах) и пр. Правила вывода могут быть представлены как ‘ЕСЛИ {множество посылок/антецедентов}, ТО {следствия} [11]. Для нашего случая сложной системы, работающей в режиме реального времени, предлагаем представление правил в следующем виде [12]:

$$\alpha_1(t_{u1})\wedge\neg\alpha_2(t_{u2})\wedge... \alpha_M(t_{uM})\rightarrow A_1(t_{A1})\wedge A_2(t_{A2})\wedge... A_N(t_{AN}). \quad (1)$$

В левой части мы видим набор логических условий α_i, связанных связкой «И» и наличием, в случае необходимости, отрицания, в правой части (следствия) записываются действия A_j, которые необходимо выполнить в данной ситуации, причем привязанные к моментам времени t_{Aj}. При этом некоторые действия способны влиять на истинность или ложность проверяемых другими правилами логических условий. Благодаря тому, что в логике управления может присутствовать несколько правил с совпадающей правой частью, но различными левыми, их можно рассматривать как одно обобщающее правило, в левой части которого левые части интегрированных правил соединяются через связку логическое «ИЛИ». В силу универсализма дизъюнктивной нормальной формы, можно сделать вывод о способности представления подобным образом любых сложных условий, влияющих на функционирование сложной системы.

Начинать проектирование логики управления следует с определения набора системных целей (задач). При этом в связи с необходимостью функционирования сложной системы в режиме реального времени каждой из целей сопоставляется момент времени, в который она должна быть достигнута. Мы можем использовать следующее множество:

$Z = \{\ (PT_j,t_j)\ \}$ – множество системных целей (задач) с заданным временем.

Комплекс бортовых управляющих алгоритмов может быть задан как

$$UA = \{\ CA_m\ \},\ m = 1...U.$$

Затем проектирование логики управления подразумевает выполнение следующих шагов:

1. Определение отображений:

$$f_1: Z \rightarrow FA \rightarrow EU \rightarrow MD \rightarrow CL,$$

иными словами, определить переходы от множества системных задач к множеству необходимой функциональности и затем к множеству необходимого оборудования, активировать которое нужно для предоставления востребованной функциональности, их режимам работы и потребляемым ресурсам.

2. Определение отображения:

$$f_2: EU \rightarrow UA,$$

иными словами, определение отображения между элементами аппаратуры и управляющими ими программными модулями.

3. Задание временных параметров:

$$f_3: EU \rightarrow T \text{ и } UA \rightarrow T.$$

Определив перечисленные отображения, мы построим циклограмму функционирования различных устройств, входящих в состав сложной системы, мы сможем вычислить уровни потребления каждого вида ресурсов для всех сценариев, на всем интервале эксплуатации системы, как функции времени.

На заключительном шаге формируется набор кортежей

$$\{ < EU_i , UA_{ij} , t_i, l_i > \}.$$

Здесь каждый из них определяет выполнение сценария UA_i, стартующего в момент времени t_i в ситуации, определяемой логическим вектором l_i.

Логический вектор (вектор состояния системы) может выглядеть, например:

$$l = (\alpha_1 = TRUE, \alpha_2 = FALSE, \ldots \alpha_q = H),$$

где α_i – конкретный признак (флаг, логическое условие). Некоторые из логических условий могут оказаться незначимыми для выполнения конкретного сценария, в этом случае, помимо значений *‘ИСТИНА’* и *‘ЛОЖЬ’*, мы используем третье значение истинности *‘Н’* («неважно»).

Используя построенные модели, для описания функционирования устройств, входящих в состав сложной системы и управляющих ими программных модулей, определить реально возникающие в логическом пространстве и во времени наложения их выполнения. Если представить их графически, мы получим: набор планов-график (циклограмм) для каждого сценария по каждому из рассмотренных видов описания поведения системы: работы оборудования, функционирования управляющих программных модулей, потребления ресурсов. Под руководством автора были разработаны инструментальные программные средства, использующие представленные подходы к моделированию, экран одного из которых показан на рис. 3. Видно отображение одного из видов циклограмм в левой части.

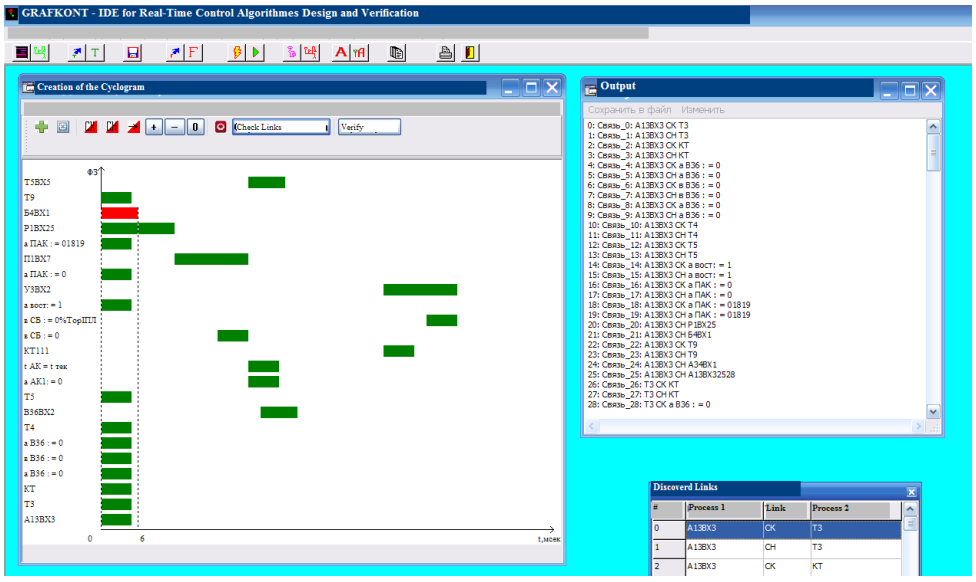

Рис. 3. Экран программного средства верификации управления сложной системой

Фактически набор планов-графиков может быть рассмотрен как набор правил логики управления сложной системой в реальном режиме времени в смысле (1). Действительно, в правой части

правил имеют места действия, привязанные к моментам времени. А левая часть (посылки) может соответствовать варианту плана-графика (циклограммы). На следующих шагах полученные правила должны найти то или иное практическое воплощение – это может быть прямое кодирование правил в исходных текстах управляющих программных модулей, автоматический синтез программ [3], использование базы знаний реального времени [7, 10] или иные подходы.

Проблема верификации логики управления сложной системой

В предыдущем разделе описан подход, позволяющий осуществлять «прямое преобразование» – отталкиваясь от целей системы, в конечном итоге получить набор правил логики управления в реальном времени. При этом актуальна и обратная задача – проверить для уже реализованных средств управления сложной системой, действительно ли они реализуют необходимую логику управления и управление, «целостное» или «согласованное» с точки зрения следующих соображений:

– соответствие имеющимся временным ограничениям, например, требованиям предшествования между выполняемыми операциями, требованиям отсутствия наложения запрещенных для этого процессов друг на друга;

– функционирование набора устройств сложной системы во всех возможных сценариях и на всем временном интервале использования без превышения доступных предельных уровней по каждому из видов ресурсов;

– обеспечение надежности, иными словами, гарантия выполнения системных задач даже в случае возникновения нештатных ситуаций и отказов каких-либо приборов и устройств.

В случае обнаружения нештатной ситуации, связанной с конкретной аппаратной проблемой, логика управления должна предусматривать проверку соответствующих логических условий, входящих в состав нужного логического вектора, и выполнение адекватных парирующих действий. При этом мы должны проверить наличие в нужные моменты времени достаточного уровня функциональности для достижения системой всех поставленных перед ней целей.

Здесь можно выделить два вида верификации: верификация логики управления как таковой, и верификация правильности ее реализации конкретными средствами (как правило, логика управления «рассредоточена» в исходном коде управляющих программных модулей).

Для выполнения второго вида верификации необходимо сначала провести анализ управляющих программных модулей – «извлечение» из них правил логики управления, как в популярном сейчас подходе software model checking [13]. Для этого можно провести синтаксический анализ исходных текстов управляющих программных модулей с выделением управляющих конструкций, связанных с переключением режимов аппаратуры, передачей управления другим программам и выполнения проверок. Под руководством автора планируется разработка подобных инструментальных программных средств.

Для проверки с представленных точек зрения большого набора правил логики управления, представленных в форме (1), мы можем прибегнуть к использованию одного из перспективных подходов поиска решения в условиях ограничений, известных в современных информационных технологиях. Это программирование в ограничениях (constraint programming, CP), SAT-решатели (boolean satisfiability – SAT solvers), динамическое программирование (dynamic programming – DP), SMT-решатели (Satisfiability Modulo Theories). В ряде случаев доступны на условиях свободной лицензии так называемые «решатели», использующие для поиска решений достаточно развитые и оптимизированные алгоритмы [14]. Доступ к возможностям подобных решателей возможен через использование интерфейса прикладного программирования (API), а в некоторых случаях – даже онлайн в сети Интернет.

Ответом решателя может быть «sat», удовлетворяющий поставленным требованиям/ограничениям, «unsat» – для противоположного случая.

Однако когда мы хотим использовать подобный решатель, перед нами встает следующая проблема. Каждый из них использует свой собственный язык и свои собственные средства моделирования для постановки задачи верификации. Соответственно, нам необходимо предварительно «погрузить» модели нашей предметной области в термины, используемые данным решателем (например, библиотеки smt-lib).

В работе [15] описан пример использования возможностей SMT-решателя *Z*3 для проверки выполнения ограничений синхронизации в логике управления, для чего был разработан прототип

специального программного инструмента-верификатора. Представляется, что аналогичный подход применим и перспективен и для проверки согласованности и «надежности интеллектуального управления» в смысле, представленном в настоящей статье.

Заключение

В статье представлены математические модели для описания логики управления сложными техническими комплексами в режиме реального времени и в условиях непредсказуемой внешней среды. Дискутируется смысл понятия «надежного» и «интеллектуального» управления с учетом ограниченности доступных ресурсов и допустимых выбросов при необходимости выполнения всех поставленных перед системой задач, несмотря на возможные отказы аппаратуры.

В настоящее время автор координирует разработку специальных программных инструментов, позволяющих осуществлять верификацию существующей логики управления, заложенной в управляющем программном обеспечении.

Для решения задачи синтеза и верификации подобной логики с учетом ограничений по ресурсам и гарантий выполнения целевых задач, мы предполагаем в будущем задействовать потенциал SMT-решателей (Satisfiability Model Theories Solvers) [15].

Библиографический список

1. *Мостовой, Я. А.* Лекции по управлению сложными техническими системами : учеб. пособие / Я. А. Мостовой. – Самара : ФГБОУ ВПО ПГУТИ, 2014. – 192 с.
2. *Хартов, В. В.* Автономное управление космическими аппаратами связи, ретрансляции и навигации / В. В. Хартов // Авиакосмическое приборостроение. – 2006. – № 6. – С. 29–33.
3. *Калентьев, А. А.* ИПИ/CALS технологии в жизненном цикле комплексных программ управления / А. А. Калентьев, А. А. Тюгашев. – Самара : Изд-во Самарского научного центра РАН, 2006.
4. *Колташев, А. А.* Эффективная технология управления циклом жизни бортового программного обеспечения спутников связи и навигации / А. А. Колташев // Авиакосмическое приборостроение. – 2006. – № 12. – С. 20–25.
5. *Тюгашев, А. А.* Интегрированная среда для проектирования управляющих алгоритмов реального времени / А. А. Тюгашев // Известия Российской академии наук. Теория и системы управления. – 2006. – № 2. – С. 128–141.
6. *Кирилин, А. Н.* Методы обеспечения живучести низкоорбитальных автоматических КА зондирования Земли: математические модели, компьютерные технологии / А. Н. Кирилин, Р. Н. Ахметов, А. В. Соллогуб, В. П. Макаров. – Москва : Машиностроение, 2010.
7. *Ахметов, Р. Н.* Проблемы реинжиниринга автоматических космических аппаратов в аномальных полетных ситуациях и пути их решения на основе базы знаний / Р. Н. Ахметов, В. П. Макаров, А. В. Соллогуб // Вестник Самарского государственного аэрокосмического университета. – 2014. – № 1(43). – С. 9–20.
8. *Тюгашев, А. А.* Подход к обеспечению отказоустойчивости космических аппаратов на основе автоматизации проектирования интеллектуальных бортовых программных средств / А. А. Тюгашев // Надежность и качество сложных систем. – 2016. – № 2 (14). – С. 9–16.
9. *Калентьев, А. А.* Разработка информационной поддержки процесса проектирования управляющих алгоритмов бортовых комплексов управления космических аппаратов / А. А. Калентьев, Ю. М. Сыгуров // Вестник СГАУ. – 2010. – № 21 (1). – С. 58–62.
10. *Tyugashev, A. A.* Visual Builder of Rules for Spacecraft Onboard Real-Time Knowledge Base / A. A. Tyugashev // Proc. 8th KES International Conference on Intelligent Decision Technologies (KES-IDT 2016). – Tenerife, Spain, 2016. – Part II. – P. 189–205.
11. *Тюгашев, А. А.* Логическое исчисление управляющих алгоритмов / А. А. Тюгашев, А. Ю. Богатов // Труды Международного симпозиума Надежность и качество. – 2013. – Т. 1. – С. 181–193.
12. *Тюгашев, А. А.* Использование расписаний при моделировании семантики управляющих алгоритмов реального времени / А. А. Тюгашев, А. Ю. Богатов, А. В. Шулындин // Труды Международного симпозиума Надежность и качество. – 2011. – Т. 1. – С. 90–93.
13. *Holzmann, G. J.* Software model checking: Extracting verification models from source code / G. J. Holzmann, M. H. Smith // Software Testing, Verification and Reliability. – 2001. – Vol. 11, № 2. – P. 65–79.
14. Garcia, M. The Future of Optimization Technology Constraints / M. Garcia, P. Stuckey, V. H. Pascal, M. Wallace // International Symposium on Intelligent and Distributed Computing IDC 2014. – 2014. – Vol. 19. – P. 126–138.
15. *Tyugashev, A. A.* Application of SMT solvers for evaluation of Real-Time control logic of spacecraft / A. A. Tyugashev // Journal of Physics: Conference Series. – 2018. – № 1096 (1). – P. 012156.

References

1. Mostovoy Ya. A. *Lektsii po upravleniyu slozhnymi tekhnicheskimi sistemami: ucheb. posobie* [Lectures on managing complex technical systems: textbook]. Samara: FGBOU VPO PGUTI, 2014, 192 p. [In Russian]
2. Khartov V. V. *Aviakosmicheskoe priborostroenie* [Aerospace instrument-making]. 2006, no. 6, pp. 29–33. [In Russian]
3. Kalent'ev A. A., Tyugashev A. A. *IPI/CALS tekhnologii v zhiznennom tsikle kompleksnykh programm upravleniya* [IPI / CALS technologies in the life cycle of integrated management programs]. Samara: Izd-vo Samarskogo nauchnogo tsentra RAN, 2006. [In Russian]
4. Koltashev A. A. *Aviakosmicheskoe priborostroenie* [Aerospace instrument-making]. 2006, no. 12, pp. 20–25. [In Russian]
5. Tyugashev A. A. *Izvestiya Rossiyskoy akademii nauk. Teoriya i sistemy upravleniya* [Proceedings of the Russian Academy of Sciences. Theory and control systems]. 2006, no. 2, pp. 128–141. [In Russian]
6. Kirilin A. N., Akhmetov R. N., Sollogub A. V., Makarov V. P. *Metody obespecheniya zhivuchesti nizkoorbital'nykh avtomaticheskikh KA zondirovaniya Zemli: matematicheskie modeli, komp'yuternye tekhnologii* [Methods for ensuring the survivability of low-orbit automatic earth sensing spacecraft: mathematical models, computer technologies]. Moscow: Mashinostroenie, 2010. [In Russian]
7. Akhmetov R. N., Makarov V. P., Sollogub A. V. *Vestnik Samarskogo gosudarstvennogo aerokosmicheskogo universiteta* [Bulletin of the Samara state aerospace University]. 2014, no. 1 (43), pp. 9–20. [In Russian]
8. Tyugashev A. A. *Nadezhnost' i kachestvo slozhnykh sistem* [Reliability and quality of complex systems]. 2016, no. 2 (14), pp. 9–16. [In Russian]
9. Kalent'ev A. A., Sygurov Yu. M. *Vestnik SGAU* [SGAU Bulletin]. 2010, no. 21 (1), pp. 58–62. [In Russian]
10. Tyugashev A. A. *Proc. 8th KES International Conference on Intelligent Decision Technologies (KES-IDT 2016)*. Tenerife, Spain, 2016, part II, pp. 189–205.
11. Tyugashev A. A., Bogatov A. Yu. *Trudy Mezhdunarodnogo simpoziuma Nadezhnost' i kachestvo* [Proceedings of the International Symposium Reliability and Quality]. 2013, vol. 1, pp. 181–193. [In Russian]
12. Tyugashev A. A., Bogatov A. Yu., Shulyndin A. V. *Trudy Mezhdunarodnogo simpoziuma Nadezhnost' i kachestvo* [Proceedings of the International Symposium Reliability and Quality]. 2011, vol. 1, pp. 90–93. [In Russian]
13. Holzmann G. J., Smith M. H. *Software Testing, Verification and Reliability*. 2001, vol. 11, no. 2, pp. 65–79.
14. Garcia M., Stuckey P., Pascal V. H., Wallace M. *International Symposium on Intelligent and Distributed Computing IDC 2014*. 2014, vol. 19, pp. 126–138.
15. Tyugashev A. A. *Journal of Physics: Conference Series*. 2018, no. 1096 (1), p. 012156.

Тюгашев Андрей Александрович
доктор технических наук, профессор,
кафедра вычислительной техники,
Самарский государственный
технический университет
(Россия, г. Самара, ул. Молодогвардейская, 244)
E-mail: tau797@mail.ru

Tyugashev Andrey Aleksandrovich
doctor of technical sciences, professor,
sub-department of computer engineering,
Samara State Technical University
(244 Molodogvardeyskaya street, Samara, Russia)

Образец цитирования:

Тюгашев, А. А. Проблемы надежного интеллектуального управления сложными системами в режиме реального времени / А. А. Тюгашев // Надежность и качество сложных систем. – 2020. – № 3 (31). – С. 28–36. – DOI 10.21685/2307-4205-2020-3-4.